Dr. Yihsiang Liow (February 16, 2017)

Contents

- http://en.wikipedia.org/wiki/Elevator
- http://science.howstuffworks.com/transport/engines-equipment/elevator1.htm

1 Requirements

Assume a discrete time system where time is incremented by 0.1s.

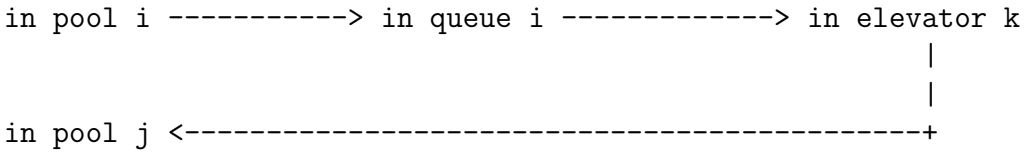
Here's the big picture:

- A user is someone using the elevator.
- A pool is a container for users. There is one for each floor. This simulates for instance users working on a floor.
- On each floor, there's a queue for each elevator.

More details:

- Users

- There's a fixed total number of users. Initially all users are in the pool at floor 0. (See Pool.)
- Each user can keep track of the start time of queueing and end time of reaching his/her destination floor.
- Each user randomly decides to join a queue for an elevator ride.
- Each user can decide is he/she enters an elevator only when its going in his/her desired direction.
- Each user will only request for a destination floor that is different from his/her current floor.
- Attributes:
  - floor (note that if the user is in the elevator, the floor value is the value of the floor which he/she left – it will be updated only when he/she enters a different floor by leaving the elevator.)
  - state: in pool, in queue, in elevator
  - probability of deciding to enter elevator when it's in the wrong direction
  - probability of entering queue
  - method to determine destination floor
  - start time
  - end time
  - total time
  - (Keep total list of activities?)
- State diagram:



- Number of elevators
- Number of floors
- Queue: At each floor, there's a queue for users waiting to enter any opened elevator. (Note that there's only one queue on each floor). Initially all queues are empty.
  - A user can enqueue the queue at that floor.
  - Any user in the queue can leave the queue (not just the front). This is to simulate the fact that a user wants to go up but the elevator is going down and therefore the user can choose not to enter the elevator.
- Pool: At each floor, there's a container for users. This simulates for instance people working at that floor Initially all containers are empty except for the container at floor 0.

- A user can leave a pool and join a queue (by enqueue) at the same floor. At that point the user must issue a request to go up/down in an elevator.

- A user can leave a queue (by dequeue) and enter an elevator when the elevator is at that floor, has stopped, door is opened, and the resulting weight has not exceeded the capacity of the elevator. [What about direction? Can a user request for down but enter if the elevator is going up? Should there be two queues, an up and a down queue?]

- A user can make one request for a floor while in the elevator (once?)

- A user must leave the elevator when the elevator has reached the user's requested floor, the elevator has stopped, and the door is open. When the user leaves the elevator, he/she enters the pool at that floor.

- Elevators:
  - can go up and down, open/close doors when they arrive at a floor.
  - Note that an elevator receives a command to go from floor to floor. The elevator does not receive a list of commands.
  - This is the same for opening and closing the door.
  - In other words, once an elevator receives a command, it acts and will only begin receiving a command when the current one is completed. This is the case except for an emergency command (interrupt) which is not implemented yet.
  - For each elevator there's a direction light for every floor telling the user which direction of the elevator. [WARNING: This light can be "up" even when the elevator is temporarily stationary.]
  - The elevator also has an direction indicator. The master can turn in to "up" or "down" or "stationary" (To the user, this corresponds to an lighted up arrow, lighted down arrow, and up and down arrow both off.)

- On each floor, there are two buttons, up and down. Call these request buttons. When a user press a request button, the information (floor, up/down) is sent the the Master.

- In each elevator, there are buttons for each floor. Call these floor buttons. When a floor button is pressed, the information (elevator id, floor) is sent to the Master.

- Master can send data to each elevator. The data includes:

- Query for floor number where an elevator is at. (Floor only? Or include between floord?)
- Query if the door is opened or closed (Door half open, etc?)
- Query if the elevator is stationary, moving up, or moving down.
- Query for total user weight. (Or assume user will not enter is reach capacity?)
- Issue command for elevator to close or open door.
- Issue command for elevator to go to a floor.

- Performance is measure by waiting time for users. This includes waiting for the elevator to arrive and also time to reach destination. (Why? Because an smart elevator can avoid opening for a customer whieh is going up while elevator is going up while there's another elevator going in the same direction.)

```
VISUALIZATION
```

2 Controller

A controller sends commands to elevators. Each elevator system in a building has only one controller. Note that the controller has the following information:

- When a user press either the up or down button at the elevator lobby on a floor, that data is sent to the controller.
- When a user press a floor number in an elevator, that data is sent to the controller.

```
class Controller
{
private:
    bool up[N];       // up[i] is true if the up button at floor i is pressed
    bool down[N];     // up[i] is true if the up button at floor i is pressed
    bool floor[E][N]; // floor[e][i] is true if floor i of elevator e is
                      // pressed
};
```